UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/797,238	03/10/2004	Rajeev B. Rajan	MSFT-2924/306986.01	2995

41505 7590 05/29/2009

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
TIMBLIN, ROBERT M	

ART UNIT	PAPER NUMBER
2167	

MAIL DATE	DELIVERY MODE
05/29/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/797,238	RAJAN ET AL.	
	Examiner	Art Unit	
	ROBERT TIMBLIN	2167	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 March 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-12, 14-17, 19-22, and 24-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-12,14-17,19-22 and 24-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *4/7/2009*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This Office Action corresponds to application 10/797,238 filed 3/10/2004.

Information Disclosure Statement

The Information Disclosure Statement filed 4/7/2009 as been acknowledged.

Response to Amendment

In the present reply (amendments submitted 3/4/2009) Applicant has amended claims 1, 14, 19, and 24. Claims 1-12, 14-17, 19-22, and 24-35 are pending.

Claim Objections

Examiner thanks Applicant for the correcting amendments to overcome the prior objection. Accordingly, the previous claim objections have been withdrawn.

However, in light of the amended claims, claim 24 is objected to for the following informalities:

Specifically, Claim 24, page 8 recites "passing the storage platform path name." The claim is objected to because it is unclear where the platform name is passed to. In correspondence to the independent claims (e.g. claim 1), it can be interpreted that the storage platform path name is passed to a database server.

Claim 24 is further objected to because the limitations lead by "identifying" and "passing" should instead begin with "*identify*" and "*pass*" as to correct grammatical error.

Response to Arguments

Applicant's arguments filed in the reply dated 3/4/2009 (i.e. 'reply' hereafter) have been fully considered but they are not persuasive.

On page 13 of the reply, Applicant argues that Sedlar fails to teach storing a filestream field separately from a relational database table as recited in claim 1. Examiner respectfully disagrees.

As cited above, Sedlar is relied upon to teach said storing a filestream field (col. 10 line 20) separately (col. 10 lines 8-24 and fig. 7; e.g. Sedlar discloses searching an index for a target file stored in a files table 710) from a relational database table (col. 7 line 36-39).

Specifically, Sedlar teaches using a hierarchical index (col. 8 line 55, and fig. 5) to locate a target file (e.g. a BLOB or the exemplary "Example.doc" file) which may be stored in a files table (710). As such, Sedlar teaches "filestream field(s)" as indices (e.g. FileID) in an indexing table while the target entry (e.g. data for the entries) can be seen as separately stored in the files table. Because Sedlar is seen to teach storing fields in an index and the data for those fields in a separate table, Sedlar sufficiently teaches the argued limitation as claimed.

Applicant also argues (p. 13 of the reply) that Sedlar does not teach unifying the transaction and model with the file sharing model as argued in the previous amendments. Examiner respectfully disagrees and maintains that Sedlar teaches this

aspect. Specifically, Sedlar teaches a system to support transactions (e.g. title) as well as the same system supporting file sharing (e.g. col. 13 line 59-col. 14 line 22). Examiner further submits that Applicant's generic argument is not specific to any particular limitation and has been addressed in this and prior actions accordingly.

Lastly, in respect to the amended claims, Applicant states that Sedlar lacks the cited limitations (pages 13-15 of the reply) as presented. Examiner respectfully disagrees and submits, as cited below, Sedlar teaches these limitations presently presented. Furthermore, Examiner respectfully submits that said amended limitations appear to describe merely the operations of detecting and executing well-known file system statements in the event to process an item (which may be a file). Examiner asserts that Sedlar teaches operating on a file by using its location via pathname to identify the file and thus perform and execute operations in a specified file system statement. As such, Sedlar is seen to teach the claims as amended and thus the arguments are found unpersuasive.

35 USC § 101

Applicant's reply submitted 3/4/2009 is sufficient to pass method claims 1, 14, and 19 under the machine or transformation test. Accordingly, the section 101 rejection of these claims is withdrawn.

Claim 24 and depending claims are now accepted under 35 USC 101 for claiming a hardware system (i.e. a machine) including a processor. In accordance with figure 7 and Applicant's paragraph 0059, the processor is best construed as a hardware processor in a computer system (i.e. it is connected to various hardware components in a computer). The rejection has been withdrawn.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-12, 14-17, 19-22, and 24-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sedlar (U.S. Patent 6,922,708) in view of Reed et al. ('Reed' hereafter) (U.S. Patent 7,035,874 B1). In the following citations, drawings, and drawing references, Sedlar teaches:

With respect to claim 1, Sedlar teaches A method for executing a transaction comprising at least one query language statement and at least one file system statement, each statement relating to a user defined type ("UDT") associated with a database server, the method comprising:

storing at least one field (col. 7 line 36-39) relating to the UDT in a relational database table (col. 7 line 46), wherein at least one field is a filestream field (col. 10 line 20), and wherein data (co. 10 line 12; e.g. Example.doc) for each filestream field (col. 10 line 20) is stored in a respective file separate (col. 10 lines 8-24 and fig. 7; e.g. Sedlar discloses searching an index for a target file stored in a files table 710) from the relational database table (col. 7 line 36-39);

receiving each of the at least one file system statements (col. 17 line 29; e.g. transaction statement), wherein each statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) comprises a call to open a first item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) and at least one of a call to read from the item (col. 12 line 65 – reading a file) and to write to the item (col. 12 line 65 – writing to a file), and a call to close the item (col. 12 line 66 – closing a file);

receiving each of the at least one query language statement (col. 6 line 25-28; e.g. receiving a query), wherein each query language statement (col. 6 line 25-28; e.g. receiving a query) is associated with a second item (col. 11 line 8-30);

for each file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call):

upon detecting a storage platform (614) path name (col. 8 line 62; e.g. /Windows) associated with the first item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) in a file system statement (col. 12 line 65-67) forwarding the file system statement (col. 12 line 65-67) to an agent (104), wherein the agent (104) performs a call

to the storage platform (614) by passing the storage platform path name to the storage platform (614);

identifying the first item (col. 10 line 15-17) based upon the storage platform path name (col. 8 line 62; e.g. /Windows);

passing a database engine function (col. 6 line 44) that returns a file system path name (col. 8 line 67) corresponding to the first item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) to the database server (col. 6 line 43-47);

performing a table look-up (col. 9 line 15) for the UDT associated with the first item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) on the database server (col. 6 line 43-47);

extracting a real file system path (col. 8 line 62-63; e.g. “input pathname”) corresponding to the first item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) using the database engine function (col. 6 line 44);

using the real file system path (col. 8 line 62-63; e.g. “input pathname”) to perform an operation on the first item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) by passing the real file system path (col. 8 line 62-63; e.g. “input pathname”) back to the agent (104), wherein the agent (104) then interacts with the file system (fig. 6) to cause execution of file system statement (col. 12 line 65-67), wherein

if the file system statement includes open, read and close operations (col. 13 line 3-10):

creating a transaction (col. 13 line 3-4; e.g. beginning a transaction) as part of an open operation (col. 13 line 3), wherein the transaction is managed separately (col. 12 line 48-61, figure 3; e.g. OS File API) from the database server (drawing reference 204);

obtaining a read lock on a data table row (col. 14 line 1-4) associated with the associated first item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) for the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call);

performing a read operation in the context of the transaction (col. 13 line 6; e.g. read from a file);

committing the transaction as part of a close operation (col. 13 line 17-18);

if the file system statement includes open, write and close operations (col. 13 line 5-10):

creating a transaction (col. 13 line 3-4; e.g. beginning a transaction) as part of an open operation (col. 13 line 3), wherein the transaction is managed separately (col. 12 line 48-61, figure 3; e.g. OS File API) from the database server (204);

obtaining a write lock (col. 13 line 36-41) on a data table row associated with the associated first item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) for the file system statement (col. 17 line 29);

performing a write operation in the context of the transaction (col. 13 line 5);

committing the transaction as part of a close operation (col. 13 line 17-18;

for each query language statement (col. 6 line 25-28; e.g. receiving a query), starting a transaction (col. 14 line 12; operation (1)) on the database server (204) updating fields (col. 14 line 14-15; operation (5)) associated with the second item (col. 11 line 8-30) in the query language statement (col. 6 line 25-28; e.g. receiving a query) and sending an updategram (col. 14 line 5-22) to the database server (204).

Sedlar does not appear to expressly teach a user defined type ("UDT").

Reed, however, teaches a user defined type ("UDT") (col. 4 line 19, i.e. a MediaUDT) for including a user defined type field (Reed at col. 1 line 14-16).

In the same field of endeavor, (i.e. data control), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because Reed would have given Sedlar a user defined type field for a more robust system of including user defined data types. Sedlar discloses such a need of various types (col. 17 line 64).

With respect to claim 2, and similar claims 15, 20, and 24, Sedlar fails to teach wherein the data table row that includes a user defined type corresponding to the item.

Reed, however, teaches a data table row that includes a user defined type corresponding to the item (col. 4 line 19, i.e. a MediaUDT) for including a user defined type field (Reed at col. 1 line 14-16).

In the same field of endeavor, (i.e. data control), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to

combine the teachings of the cited references because Reed would have given Sedlar a user defined type field for a more robust system of including user defined data types.

With respect to claim 3, Sedlar teaches the method of claim 1, wherein each query language statement is a T-SQL statement (col. 13 line 1-10).

With respect to claim 4, Sedlar teaches the method of claim 1, wherein transactions created for the file system statements are managed by a storage platform (col. 12 line 48-61 and line 62-67, figure 3; e.g. OS File API).

With respect to claim 5, Sedlar teaches the method of claim 1, further comprising receiving a transaction context for file system operations (col. 10 line 55-56) and performing at least one of a read lock and a write lock consistent with the received transaction context (col. 13 line 7 and col. 14 line 1-5).

With respect to claim 6, Sedlar teaches the method of claim 1, wherein creating the transaction comprises:

determining whether creating the transaction will result in a conflict (col. 13 line 35-46; e.g. determining a session lock on data);

if creating the transaction will result in a conflict, then resolving the conflict according to a conflict resolution scheme (col. 13 line 36-40 and col. 14 line 1-5; e.g. waiting until a transaction commits to see changes to data or new rows); and

if creating the transaction will not result in a conflict, then starting the transaction (col. 13 line 42-46).

With respect to claim 7, Sedlar teaches the method of claim 1, wherein acquiring the read lock on the row comprises acquiring a read committed view of the row (col. 14 line 4-5).

With respect to claim 8, Sedlar teaches the method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent another transaction from accessing the row while the transaction is being processed (col. 13 line 35-47).

With respect to claim 9, Sedlar teaches the method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent a non-transacted file system statement from accessing the row while the transaction is being processed (col. 14 line 5-22).

With respect to claim 10, Sedlar teaches the method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent another statement within the transaction from writing to the row (col. 13 line 66-col. 14 line 2).

With respect to claim 11, Sedlar teaches the method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will enable another statement within the transaction to read from the row (col. 13 line 56-57).

With respect to claim 12 and similar claims 17, 22 and 35, the method of claim 1, Sedlar teaches comprising starting the transaction by acquiring one of a read lock and a write lock on a file stream field of the row (col. 10 line 8-24 and col. 11 line 41).

With respect to claim 14, Sedlar teaches A method for locking and isolation of a file system statement, the method comprising:

receiving the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) comprising a call to open an item (col. 12 line 65 – opening a file), a call to read from the item (col. 12 line 65 – reading a file) or to write to the item (col. 12 line 65 – writing to a file), and a call to close the item (col. 12 line 66 – closing a file), the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) being independent of any database commands employing a query language of a

database (col. 5 line 10-13) and wherein each item reference in a statement is stored separately (drawing reference 710) from a database table (drawing reference 510) associated with the item (col. 10 line 8-23, figs 5,7; e.g. Sedlar teaches storing a hierarchical index separately from a files table);

upon detecting a storage platform path name (col. 8 line 62; e.g. /Windows) associated with the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) in a file system statement (col. 12 line 65-67) forwarding the file system statement (col. 12 line 65-67) to an agent (104), wherein the agent (104) performs a call (col. 5 line 7) to the storage platform (614) by passing the storage platform path name (col. 8 line 62; e.g. /Windows) to the storage platform (614);

identifying the item (col. 10 line 15-17) based upon the storage platform path name (col. 8 line 62; e.g. /Windows);

passing a database engine function (col. 6 line 44) that returns a file system path name (col. 8 line 67) corresponding to the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) to a database server (col. 6 line 43-47);

performing a table look-up (col. 9 line 15) for a user defined type associated with the item on the database server (col. 6 line 43-47);

extracting a real file system path (col. 8 line 62-63; e.g. “input pathname”) corresponding to the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) using the database engine function (col. 6 line 44);

using the real file system path (col. 8 line 62-63; e.g. “input pathname”) to perform an operation on the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file), wherein:

in response to receiving the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) that is independent of any database application programming interface requests (col. 5 line 10-13), determining a read lock is available (col. 13 line 35-45) for a row of a data table corresponding to the item (col. 13 line 7 and 41-42 – i.e. locking a row associated with a file);

if the read lock is not available for the row of the data table corresponding to the item, then failing the open (col. 13 line 36-40 and col. 14 line 1-5; e.g. waiting until a transaction commits to see changes to data or new rows); and

if the read lock is available for the row of the data table corresponding to the item:

performing a shared open for the item (col. 13 lien 54-58);

acquiring a read lock on the row (col. 13 line 42-46).

Sedlar does not appear to expressly teach a user defined type (“UDT”).

Reed, however, teaches a user defined type (“UDT”) (col. 4 line 19, i.e. a MediaUDT) for including a user defined type field (Reed at col. 1 line 14-16).

In the same field of endeavor, (i.e. data control), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because Reed would have given Sedlar a

user defined type field for a more robust system of including user defined data types. Sedlar discloses such a need of various types (col. 17 line 64)

With respect to claim 16, the method of claim 14, wherein acquiring the read lock on the row comprises acquiring a read committed view of the row (col. 14 line 4-5).

With respect to claim 19, Sedlar teaches A method for locking and isolation of a file system statement, the method comprising:

receiving the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) comprising a call to open an item (col. 12 line 65 – opening a file), a call to read from the item (col. 12 line 65 – reading a file) or to write to the item (col. 12 line 65 – writing to a file), and a call to close the item (col. 12 line 66 – closing a file), the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) being independent of any database commands employing a query language of a database (col. 5 line 10-13) and wherein each item reference in a statement is stored separately (drawing reference 710) from a database table (drawing reference 510) associated with the item (col. 10 line 8-23, figs 5,7; e.g. Sedlar teaches storing a hierarchical index separately from a files table);

upon detecting a storage platform path name (col. 8 line 62; e.g. /Windows) associated with the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) in a file system statement (col. 12 line 65-67) forwarding the file system statement

(col. 12 line 65-67) to an agent (104), wherein the agent (104) performs a call to the storage platform by passing the storage platform path name (col. 8 line 62; e.g. /Windows) to the storage platform (614);

identifying the item (col. 10 line 15-17) based upon the storage platform path name (col. 8 line 62; e.g. /Windows);

passing a database engine function (col. 6 line 44) that returns a file system path name (col. 8 line 67) corresponding to the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) to a database server (col. 6 line 43-47);

performing a table look-up (col. 9 line 15) for a user defined type associated with the item on the database server (col. 6 line 43-47);

extracting a real file system path (col. 8 line 62-63; e.g. “input pathname”) corresponding to the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) using the database engine function (col. 6 line 44);

using the real file system path (col. 8 line 62-63; e.g. “input pathname”) to perform an operation on the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file), wherein:

in response to receiving the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) that is independent of any database application programming interface requests (col. 5 line 10-13), determining if a write lock is available (col. 13 line 35-45) for a row of a data table corresponding to the item (col. 13 line 7 and 41-42 – i.e. locking a row associated with a file);

if the write lock is not available for the row of the data table corresponding to the item, then failing the open (col. 13 line 36-40 and col. 14 line 1-5; e.g. waiting until a transaction commits); and

if the write lock is available for the row of the data table corresponding to the item:

performing an exclusive open for the item (col. 14 line 1-3);

acquiring the write lock on the row (col. 13 line 42-46).

Sedlar does not appear to expressly teach a user defined type ("UDT").

Reed, however, teaches a user defined type ("UDT") (col. 4 line 19, i.e. a MediaUDT) for including a user defined type field (Reed at col. 1 line 14-16).

In the same field of endeavor, (i.e. data control), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because Reed would have given Sedlar a user defined type field for a more robust system of including user defined data types. Sedlar discloses such a need of various types (col. 17 line 64)

With respect to claim 21, Sedlar teaches the method of claim 19, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent another statement from accessing the row while the statement is being processed (col. 13 line 40-45).

With respect to claim 24, Sedlar teaches A system for executing a file system statement in the context of a transaction, the file system statement including a call to open an item, one of a call to read from the item and a call to write to the item, and a call to close the item, the system comprising:

a processor (1804);

a relational data engine (figure 7 and 204) comprising a data table (710) having a row (fig. 7, Row ID) corresponding to the item (File ID);

a storage platform (figures 3 and 4, references 108 and 204) built on the relational data engine (figure 7 and 204), the storage platform (figures 3 and 4, references 108 and 204) comprising means for receiving the file system statement (figure 3), wherein each item reference in a statement is stored separately (drawing reference 710) from a database table (drawing reference 510) associated with the item (col. 10 line 8-23, figs 5,7; e.g. Sedlar teaches storing a hierarchical index separately from a files table), means for associating the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) with the transaction (col. 3 line 43-46), and means for starting the transaction by acquiring one of a read lock (col. 14 line 1-3) and a write lock (col. 12 line 65-66) on a data table row ((col. 13 line 7 and 41-42 – i.e. locking a row associated with a file) the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call) comprising a call to open (col. 12 line 65 – opening a file), a call to read from the item (col. 12 line 65 – reading a file) or to write to the item (col. 12 line 65 – writing to a file), and a call to close the item (col. 12 line 66 – closing a file), the file system statement (col. 3 line 40-45 and col. 12 line 45-50; i.e. an OS file system call)

being independent of any database commands employing a query language of a database (col. 5 line 10-13);

a file system (fig. 6), wherein the file system is adapted to:

upon detecting a storage platform path name (col. 8 line 62; e.g. /Windows) associated with the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) in the file system statement (col. 12 line 65-67), performing a call (col. 5 line 7) to a storage platform (614) by passing the storage platform path name (col. 8 line 62; e.g. /Windows) ;

identifying the item (col. 10 line 15-17) based upon the storage platform path name (col. 8 line 62; e.g. /Windows);

passing a database engine function (col. 6 line 44) that returns a file system path name (col. 8 line 67) corresponding to the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file)to the relational data engine (figure 7 and 204);

wherein the relational data engine(figure 7 and 204) is adapted to:

upon receiving the database engine function (col. 6 line 44) from the file system (fig. 6), perform a table look-up (col. 9 line 15) for an associated user defined type

extract a real file system path (col. 8 line 62-63; e.g. “input pathname”) corresponding to the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file) using the database engine function (col. 6 line 44);

pass the real file system path col. 8 line 62-63; e.g. “input pathname”) to perform an operation on the item) to the file system to cause the file system to perform the

operation on the item (col. 8 line 62 Example.doc file; and col. 12 line 65 – opening a file.

Sedlar does not appear to expressly teach a user defined type ("UDT").

Reed, however, teaches a user defined type ("UDT") (col. 4 line 19, i.e. a MediaUDT) for including a user defined type field (Reed at col. 1 line 14-16).

In the same field of endeavor, (i.e. data control), it would have been obvious to one of ordinary skill in the data processing art at the time of the present invention to combine the teachings of the cited references because Reed would have given Sedlar a user defined type field for a more robust system of including user defined data types. Sedlar discloses such a need of various types (col. 17 line 64)

With respect to claim 26, Sedlar teaches the system of claim 24, wherein the storage platform further comprises means for associating a second statement with the transaction (col. 13 line 59-64).

With respect to claim 27, Sedlar teaches the system of claim 26, wherein the second statement is another file system statement (col. 13 line 65-67).

With respect to claim 28, Sedlar teaches the system of claim 26, wherein the second statement is a transactional query language statement (col. 14 line 10-20).

With respect to claim 29, Sedlar teaches the system of claim 24, wherein the means for starting the transaction comprises means for performing the following steps:

determining whether starting the transaction will result in a conflict (col. 13 line 35-46; e.g. determining a session lock on data);

if starting the transaction will result in a conflict, then resolving the conflict according to a conflict resolution scheme (col. 13 line 36-40 and col. 14 line 1-5; e.g. waiting until a transaction commits to see changes to data or new rows); and

if starting the transaction will not result in a conflict, then starting the transaction (col. 13 line 42-46).

With respect to claim 30, Sedlar teaches the system of claim 24, wherein the read lock provides a read committed view of the row (col. 14 line 4-5).

With respect to claim 31, Sedlar teaches the system of claim 24, wherein the write lock prevents another transaction from accessing the row while the transaction is being processed (col. 13 line 35-47).

With respect to claim 32, Sedlar teaches the system of claim 24, wherein the write lock prevents a non-transacted file system statement from accessing the row while the transaction is being processed (col. 14 line 5-22).

With respect to claim 33, Sedlar teaches the system of claim 24, wherein the write lock prevents another statement within the transaction from writing to the row (col. 13 line 66-col. 14 line 2).

With respect to claim 34, Sedlar teaches the system of claim 24, wherein the write lock enables another statement within the transaction to read from the row (col. 13 line 56-57).

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Contact Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Robert M. Timblin whose telephone number is 571-272-5627. The examiner can normally be reached on M-Th 8:00-4:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John R. Cottingham can be reached on 571-272-7079. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/ROBERT TIMBLIN/

Examiner, Art Unit 2167

/John R. Cottingham/

Supervisory Patent Examiner, Art Unit 2167